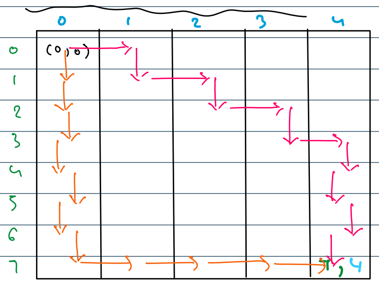
Grid Path finder



move → right (horizontally)
move → down (vertically)

V H V H V H V H H H H H

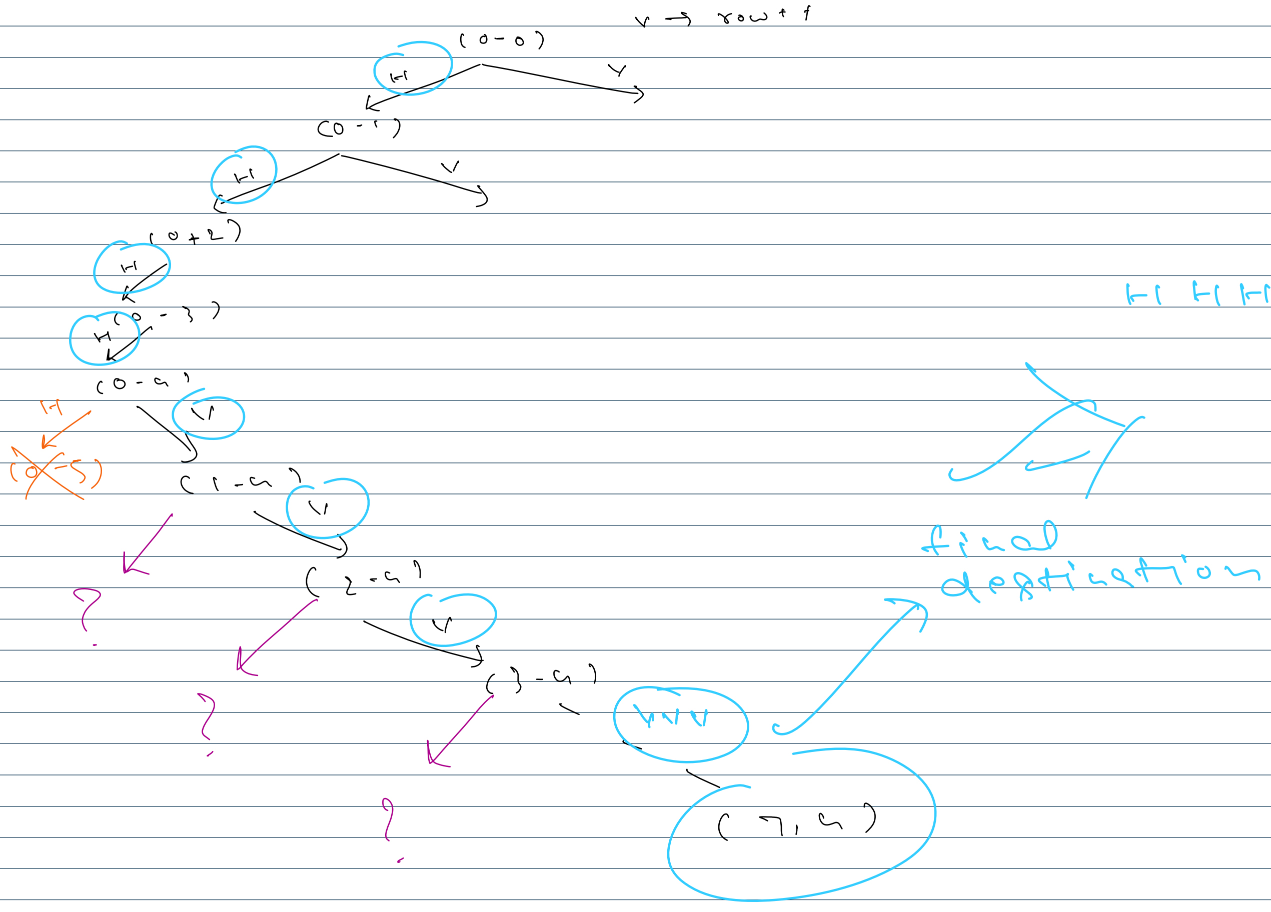
H H H H H H H H V V V V

H and V ↓ should be swapped → permission

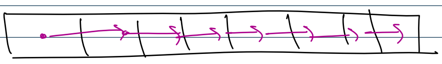
options → movements → right / down → choices → decision → Recursion

H → col+1

V → row+1



Grid 1/



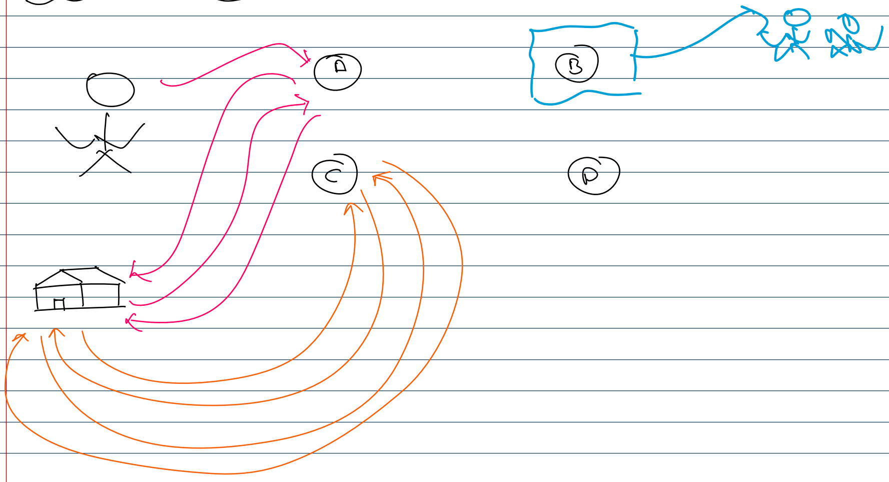
H H H H H H H H H

```
public class Main {
    public static void main(String[] args) {
        int n = 1; // rows -> r
        int m = 7; // cols -> c

        int cr = 0; // current row
        int cc = 0; // current col

        String ans = "";
        paths(n-1, m-1, cr, cc, ans);
    }

    public static void paths(int r, int c, int cr, int cc, String ans){
        if(r==cr && c==cc){
            System.out.println(ans + " STOP");
            return;
```

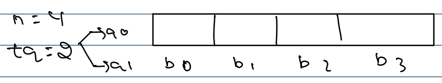
v v v v v

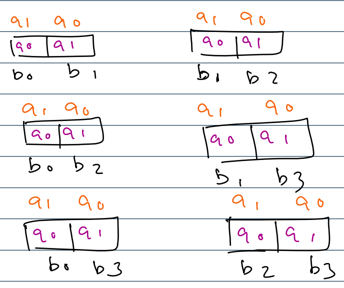
```
        }
        if(cr > r || cc > c){
            return;
        }
        paths(r, c, cr, cc+1, ans + " H ->");
        paths(r, c, cr+1, cc, ans + " V ->");
    }
}
```

Backtracking



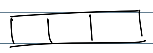
Queen Permutations → arrangements matter





Combinations = $^4C_2$ = 6

Permutations = $^4P_2$ = 12



combinations × 2!

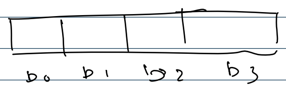
permutations

A B C

A B C
B A C
C A B
A C B
B C A
C B A

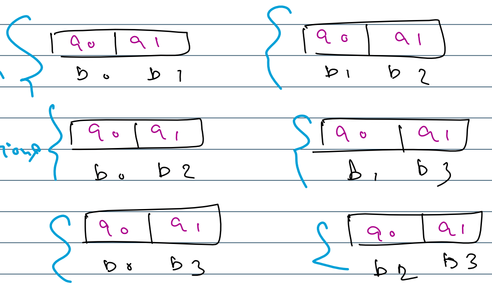
⇒ 3!

n!



queens → q0, q1

Combinations



⇒ 6

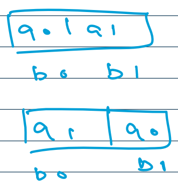
⇓

Permutations

⇓

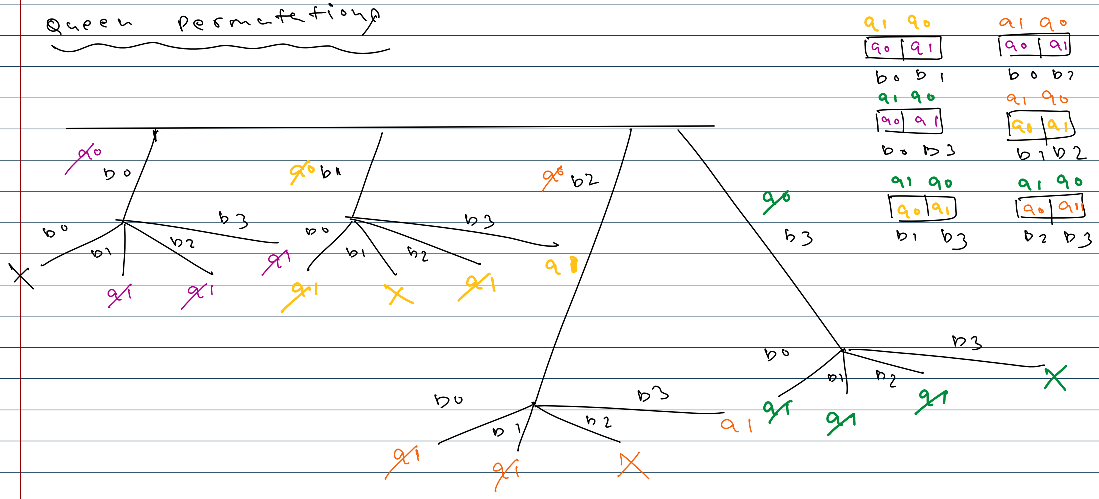
6 * 2!

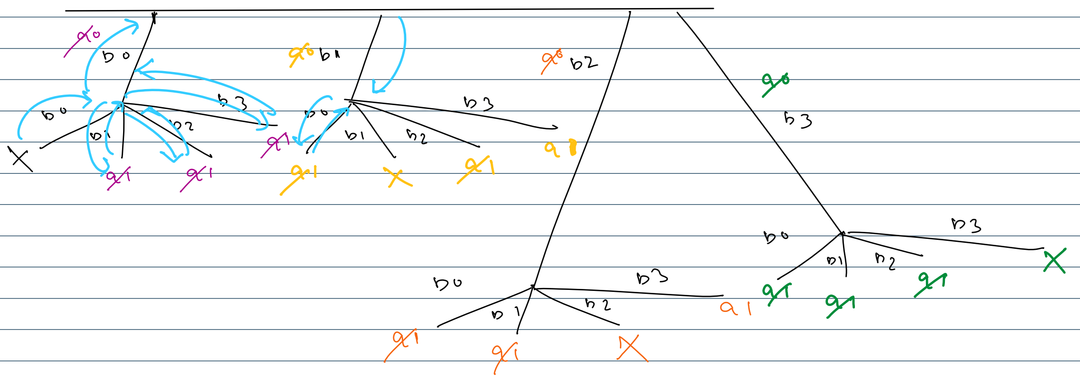
2 permutations



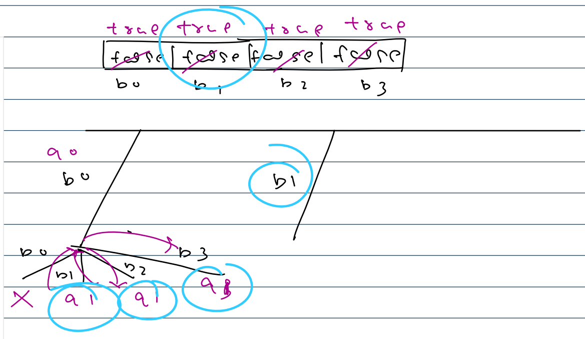
$\boxed{A|B|C} \Rightarrow$ ABC, ACB, BAC, BCA, CAB, CBA $\Rightarrow 6 \Rightarrow n!$

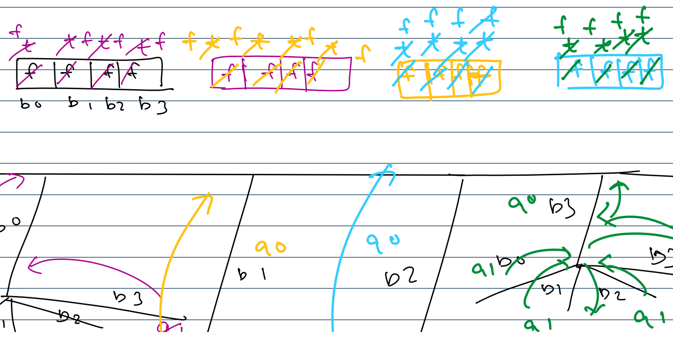
Queen permutations

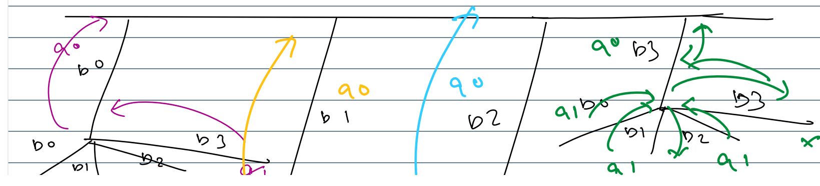




q0b0 q1b1
q0b0 q1b2
q0b0 q1b3

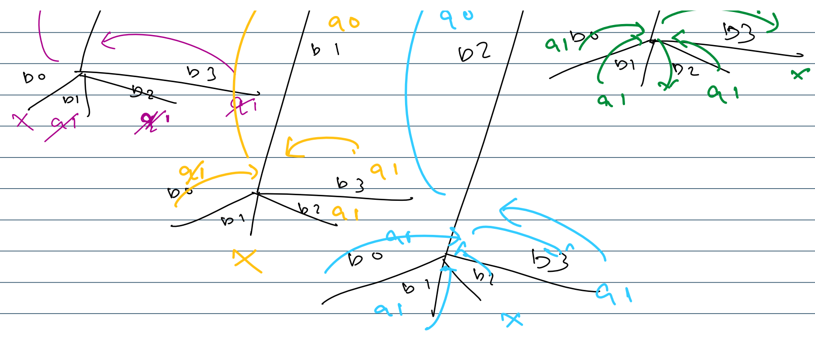






```
public class Main {
    public static void ma
        int n = 4;
        int tq = 2; // tot
        int qpsf = 0; // c
        boolean[] board =
        String ans = "";
        perm(board, tq, c
    }

    public static void pe
        if(qpsf == tq){
            System.out.pr
            return;
        }
        for(int i=0; i<boa
            if(board[i] ==
                board[i] =
                perm(boar
                board[i] =
            }
        }
```

```
ain(String[] args) {

tal queens
queen placed so far
= new boolean[n];

qpsf, ans);


rm(boolean[] board, int tq, int qpsf, String ans){

intln(ans);


rd.length; i++){
 false){
 true;
d, tq, qpsf+1, ans+ "b"+ i+ "q" + qpsf+" ");
 false;
```



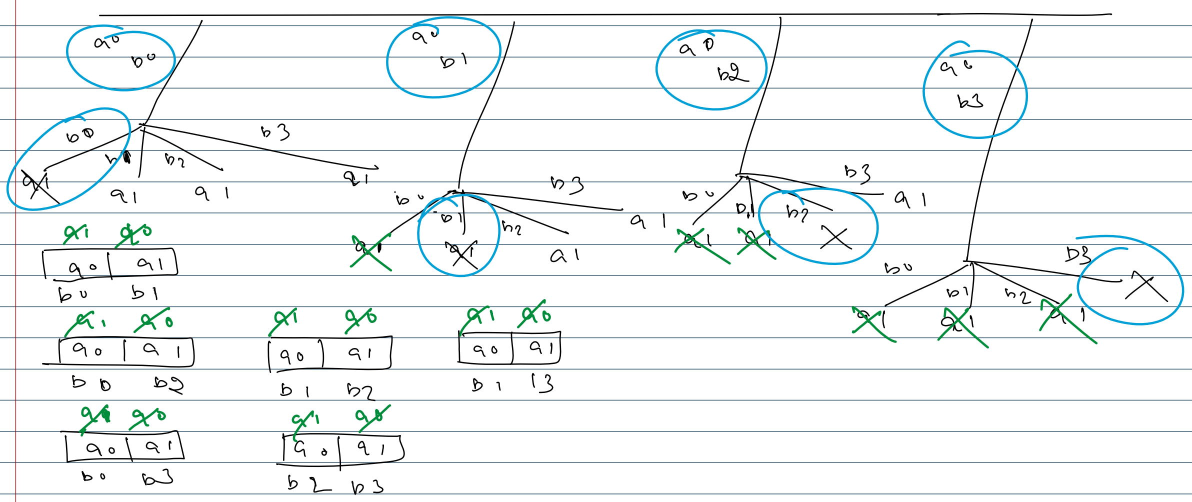
```
board[i] =
        }
    }

        }
    }
```

Combinations

q0 | q1 — b0 b1	q0 | q1 — b1 b2
q0 | q1 — b0 b2	q0 | q1 — b1 b3
q0 | q1 — b0 b3	q0 | q1 — b2 b3

Queen combinations

n : 4
q = 2

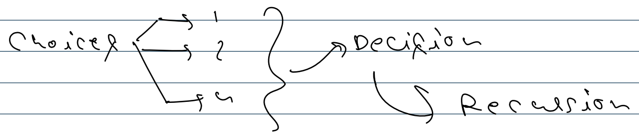


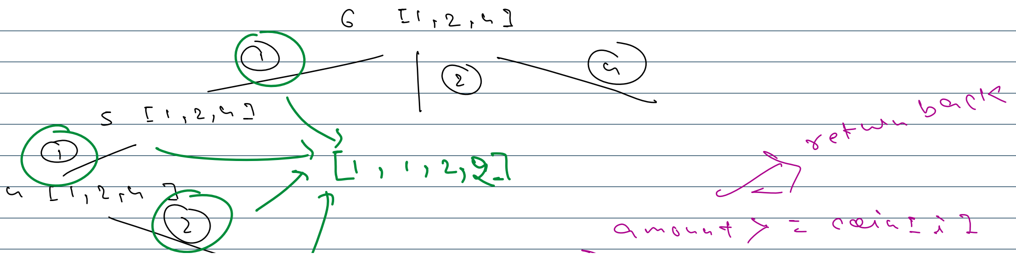
```
public class Main {
    public static void main(String[
        int n = 4;
        int tq = 2; // total queens
        int qpsf = 0; // queen plac
        boolean[] board = new bool
        String ans = "";
        int idx = 0;
        perm(board, tq, qpsf, ans,
    }

    public static void perm(boolea
        if(qpsf == tq){
            System.out.println(ans);
            return;
        }
        for(int i=idx; i<board.length
            if(board[i] == false){
                board[i] = true;
                perm(board, tq, qpsf
                board[i] = false;
            }
        }

    }
}
```

Coin change - Permutations

coin : [1, 2, 4]
amount : 6

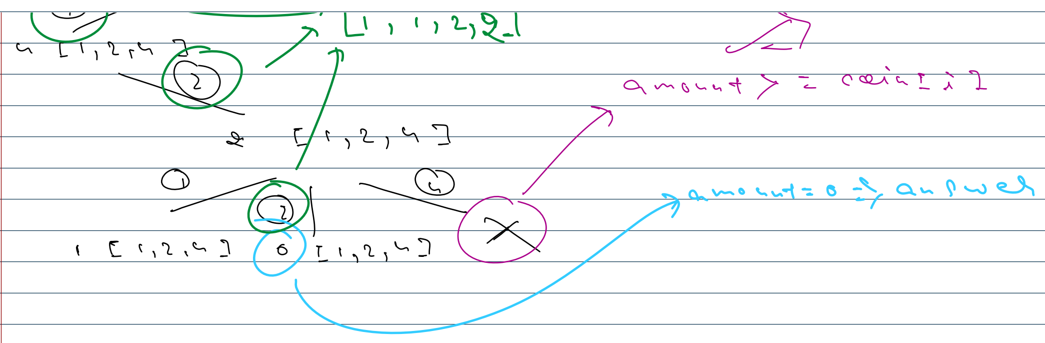
2 + 4 = 6
1 + 1 + 2 + 2 = 6
1 + 1 + 4 = 6
1 + 1 + 1 + 1 + 2 = 6

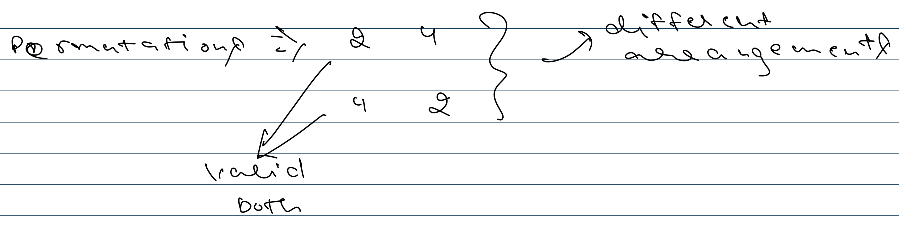




```
false;
```

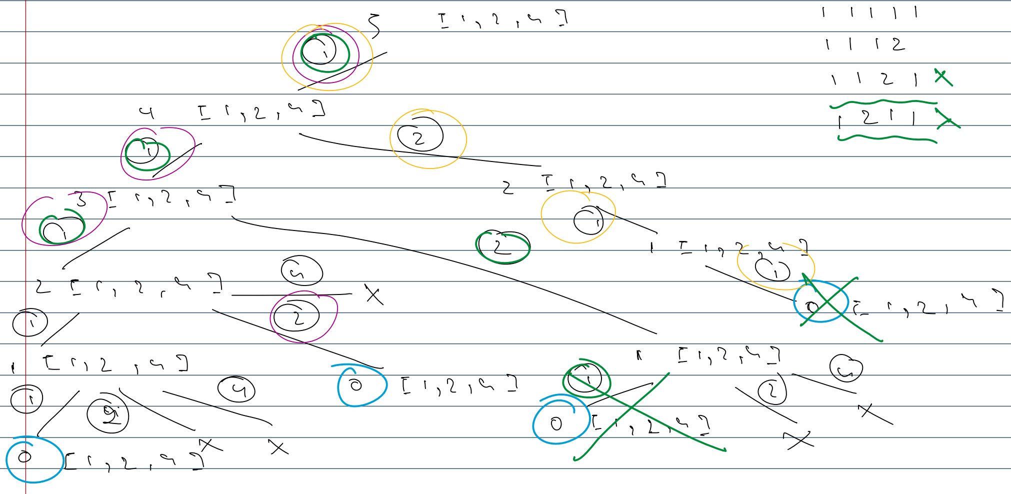
```
] args) {

ed so far
ean[n];

idx);

n[] board, int tq, int qpsf, String ans, int idx){

; i++){

+1, ans+ "b"+ i+ "q" + qpsf+" ", i+1);
```





```
public class Main {
    public static void main(String[] args) {
        int[] coins = {1, 2, 4};
        int amount = 6;
        String ans = "";
        perm(coins, amount, ans);
    }

    public static void perm(int[] coins, int amount, String ans){
        if(amount==0){
            System.out.println(ans);
            return;
        }

        for(int i=0; i<coins.length; i++){
            if(amount >= coins[i]){
                perm(coins, amount - coins[i], ans+" "+coins[i]);
            }
        }

    }
}
```

Coin change - combinations



```
public class Main {
    public static void main(String[] args) {
        int[] coins = {1, 2, 4};
        int amount = 5;
        String ans = "";
        int idx = 0;
```

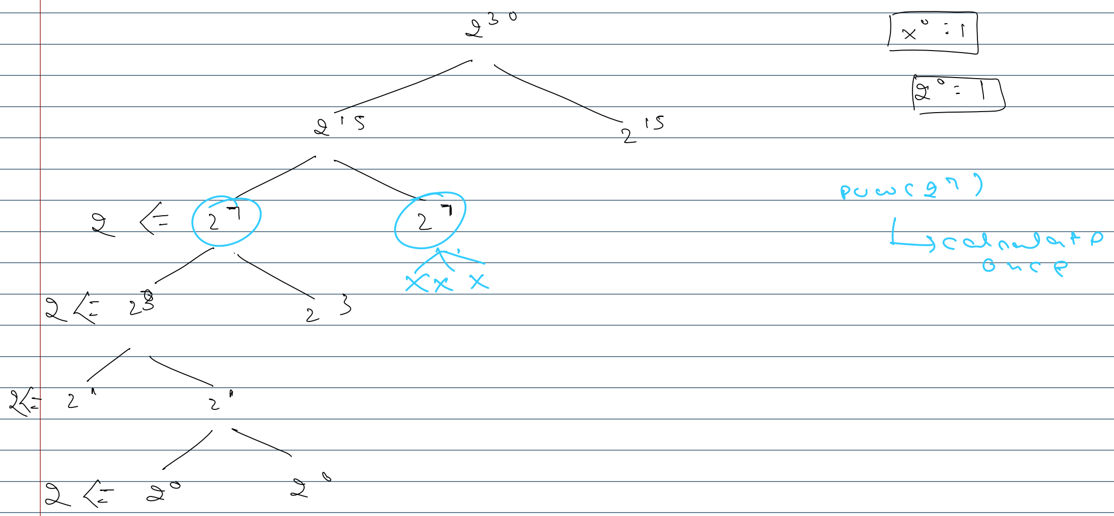
```
        perm(coins, amount, ans, idx);
    }

    public static void perm(int[] coins, int amount, String ans, int idx){
        if(amount==0){
            System.out.println(ans);
            return;
        }

        for(int i=idx; i<coins.length; i++){
            if(amount >= coins[i]){
                perm(coins, amount - coins[i], ans+" "+coins[i], i);
            }
        }

    }
}
```

Power function

$2^5 = 2\times2\times2\times2\times2 = 32$

pow(2, 5)

$2^{30} = 2\times2\times2\times\ldots2 = ?$



```
public class Main {
    public static void main(String[] args) {
        int a = 2;
        int b = 10;
        System.out.println(pow(a, b));
        System.out.println(pow1(a, b));
    }

    public static int pow(int a, int b){
        if(b==0){
            return 1;
        }
        if(b%2==0){
            return pow(a, b/2)*pow(a, b/2);
        }
        else{
            return 2*pow(a, b/2)*pow(a, b/2);
        }
    }

    public static int pow1(int a, int b){
        if(b==0){
            return 1;
        }
```

```
        int half = pow1(a, b/2);
        if(b%2==0){
            return half*half;
        }
        else{
            return 2*half*half;
        }
    }
}
```